Measurement and analysis of web tracking on the Internet

Samy Sidhoum

Supervisor:
Pere Barlet Ros

2 July 2019

ABSTRACT

"What gets measured gets managed"
Peter Drucker

When surfing on the internet, every click is recorded and analyzed in order to draw a user profile up and adapt contents and advertisements to his behavior. That is what **web trackers** are for.

Studies have been carried out to determine the scale of this phenomena, to extract and analysis **web tracker** code and to create tools to control and block abusive code.

This project focuses on the evolution of web trackers over the years. How the technology evolve ? Does **legislation** impact their use ?

The goal is to gather and **visualize** data relative to the changes in the Javascript code constituting the web trackers over time. This project goal is to develop tools retrieving source code of multiple websites at different time and **visualizing** it through an iterative interface. This project comes as a feature within the scope of the framework developed by the UPC Web tracking team to analyse obfuscated code implemented in **Python**.

Keywords: Web tracker, Visualization, Legislation, Python

ACKNOWLEDGMENTS

I would like to thank Pere Barlet Ros for his comments and guidance during this project.

To Ismael who solved some thorny problems and give a basis to my work.

To all the Webtracking team, and in particular Josep, Jeremy and Mohammed for their feedback during our meeting.

To Ines Cotton for her patient proofreading and her support.

CONTENTS

INTRODUCTION

Almost every important website gathers user data to get more information about user habits, to improve their experience or to use it for commercial purposes. As data are resources belonging to the user, the way websites and company use them must be supervised to make sure no misuse is being made. Data can be weaponized and lead to serious hazards. Cambridge Analytica gate or Yahoo's data breach proved that an ethic and reasoned use of data should be implemented. Even if some juridical blanks remain, Europe is the first world power to put a legal frame on the use of these data with the General Data Protection Regulation (GDPR). This project is supposed to be an additional module linked to a previous work on detecting web trackers. The goal of this project is to gather information on how websites are using user data. Detecting when a website doesn't seem to respect the GDPR. Having an history of how and when websites source code changed seems a mine of information. Everyone who want to check on the behavior of a company website may be able to visualize information contained in the visible part of the website source code. This project objectives were to gather source code of 2600+ websites at different time and build a visualization tool in order to discover patterns in the observed data.

Motivation:

Having interest in the topic of the cybersecurity and more generally about the ethics in computer science, the subject fits me. What I want after finishing my studies is to work on inspiring projects using the python tech which I have tried to improve for 2 years now. Discovering fundamentals libraries such as **scrapy**, **beautifulsoup, difflib, pandas, bokeh, flask**, **cprofiler** are really important for a python engineer.

By version, we design the HTML/JS source code of a website at a specific date.

All the source code is available on :
https://drive.google.com/file/d/1AA_hjzEq6MuovNA_9tVOnF3c9MuimFb_/view?usp=sharing

BACKGROUND

The purpose of this project is to be linked to a PhD thesis project on web tracking. His project is to build a web scraper, finding patterns between similar websites with an emphasis on web tracking code contained in JavaScript (JS) code. A 8-members team whose goal was to work on web tracking has been built. Everyone project was related to web tracking with more or less link to the thesis project. My project focuses directly on the thesis, creating an additional module which can be coupled to this work.

What is Web Tracking ?

According to Wikipedia, **Web tracking** is the practice by which operators of websites collect and share information about a particular user's activity on the World Wide Web. Analysis of an individual user's behaviour may be used to provide content that relates to their implied preferences.

Commercial Utility

Who use it ?

Webtracking is mainly used by companies to increase business opportunities. Companies using webtracking can be separated in two categories, first-party and third-parties. [1]

When a user visits a website, the website is the first-party. If this website embed scripts collecting the user information it shares with others actors, these actors constitute the third-parties. It is worth to mention than, once a third-party joins a website, it can invite as many other third-parties as he wants. Personal data collected and originally shared only with the first-party can be obtained by any organization.

Why do they use it ?

The main use of web tracking for commercial purposes [2][3] is to:

- Detect opportunities to get information unavailable by web form
- Identify website visitors
- Observe navigation behavior on the website
- Analyze links clicked, visited pages
- Measure time spent on every page
- Qualify hobbies by gathering information
- Carry out statistics on visits, interests and navigation
- Separate different marketing targets
- Give a mark to every visitor according to their maturity in the process of purchasing
- Give the visitor personal contents and ads
- Streamline marketing and commercial efforts

Problems raised

Ethical issues

First think is, data retrieved by web trackers are not anonymized, it is easy to match these data with the individuals, either by pairing it with their facebook profiles, either by using statistical algorithms which match the browser histories with social media profiles.

This can lead to some abuse [4] as:

- **Price discrimination** [5]:
  The webtracking statistical analysis tools allow a company to know what is the maximum value a customer would pay for a product (number of visits on the page, steps realized toward the purchase of the product). Once this information is acquired, the company can increase the product price and transform the consumer surplus into revenue. Gender, ethnicity and location can be criterias to apply price discrimination too.

- **Mass surveillances** [6] :
  Several tools using web tracking are used by governments to monitor people actions on the Internet. Revelations has been made that NSA has access to databases of GAFA companies. Multiple programs to analyse these informations such as PRISM, Xkeyscore or ECHELON has been created for mass surveillance. Many of these systems has been sold to foreign governments.

Now the question is, should we or should we not give away personal information ? There are big debates about it but a consensus agree on giving that choice to the user. That means it is necessary to inform the user, and to get his consent to carry any kind of data collect. Some websites ignore this process and take people data without their consent to sell it to the highest bidder even though legislation forbid it. Therefore, It is then necessary to have tools to prevent this kind of behavior - "*Tracking the trackers*"

Slow down the browser

The more web trackers a website page has, the longer it will take to load it. In addition of providing a bad user experience, this additional loading time can cost millions of dollars to company.
1 second of delay to load a webpage decreases the user satisfaction by 16%, 54% of user will quit a webpage if it takes 3+ seconds to load [7]

For a company winning 100.000$ by day, 1 second of delay is a loss of about 2.5 millions of dollars.

Web Tracking methods

Web tracking is above all the recognition of a user when he connects and surfs on a website. Each time a customer connects, the website must recognize him and associate his previous connection to his profile in order to understand his behavior. The goal is to encourage him to stay the longer and to offer him content or ads based on his detected interests. Several ways exists to track data online.

Here some of the most popular techniques of webtracking but the list is far from being exhaustive [8] :

Cookies:

Cookie is the first and still most used web tracking method. When a user visits a website for the first time, a cookie containing a unique identifier is created and stored in the user's browser storage. That way, as long as the cookie is not suppressed, the website can recognize the user and add more information on his profile [9]. Note that the cookies work only with the website using it, they're not **cross-domain**.

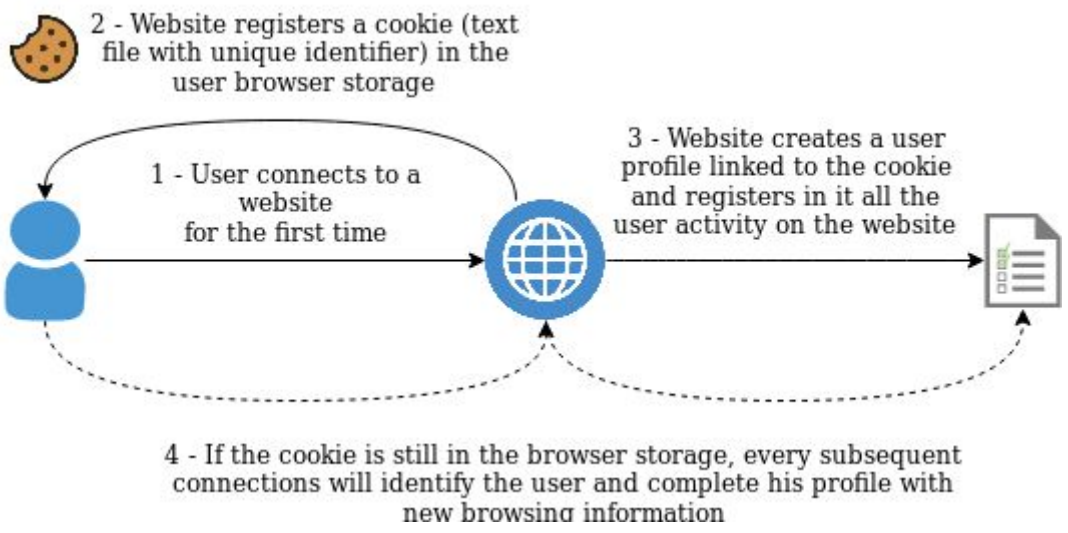

Figure 1 - How cookies work

Fingerprinting:

The fingerprinting consists in detecting the user computer configuration including browser type and version, operating system and version, screen resolution, supported fonts, plugins, time zone, language and font preferences and hardware configurations. All these information provide an accurate way to identify a user.

These information can be stored into a **canvas** HTML object, with dimensions depending on the computer configuration. This technique is called **canvas fingerprinting** .

Contrary to simple cookie, the fingerprinting technique does not require any information registered on the browser storage and can be considered more “invisible”.

A previous work, estimated at 6.4% the number of website using canvas fingerprinting among the 10.000 most popular website [10].

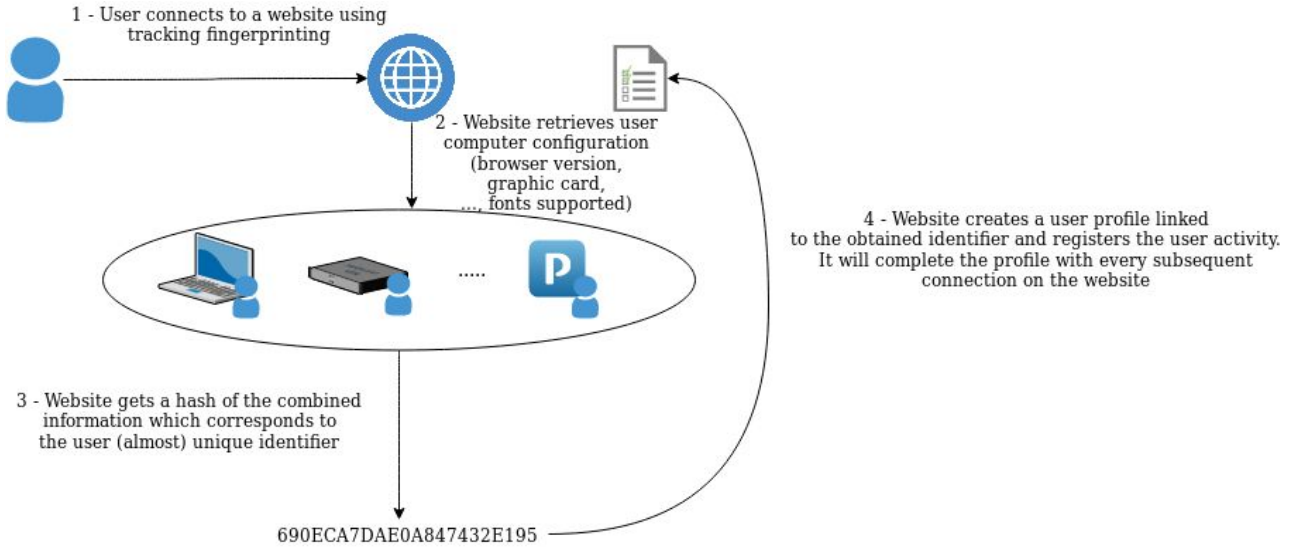

Figure 2 - Tracking fingerprinting

Web beacon [11]:

Web beacon consists in creating a picture usually invisible (size of a pixel and same color as the background) which will be downloaded when the page is load. To download the image, a request is send to a tracking server which can keep track of what content the user has accessed to. This technics is also used for email tracking: When a user open an email, it download the image which is joined and give the information that the email has been opened.

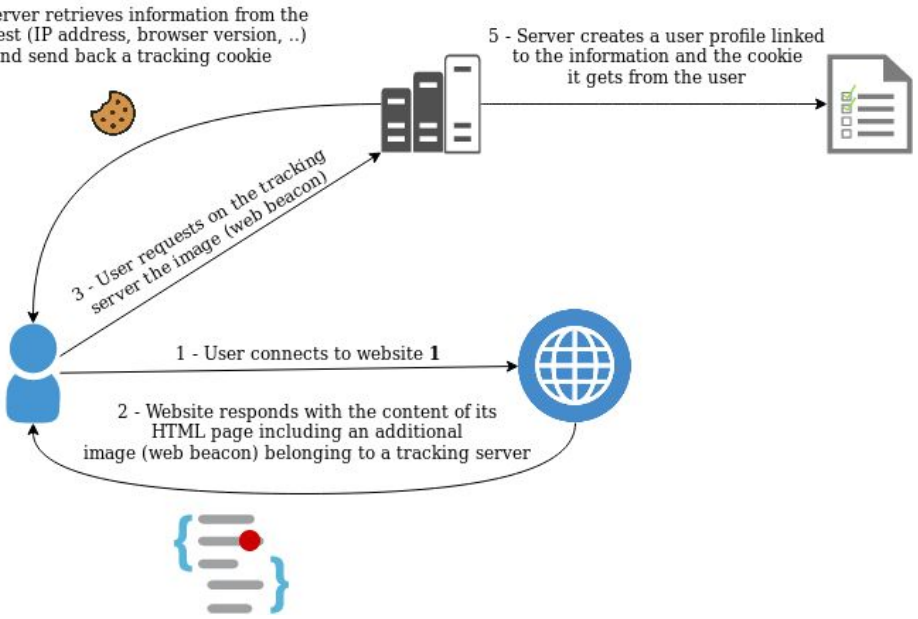

Figure 3 - Web beacon tracker - Part 1

Web beacons are very powerful tracker because they are **cross-domain**. The tracking server could have inserted a web beacon into multiple websites, which means it is able to track a user on every website where it has a beacon.

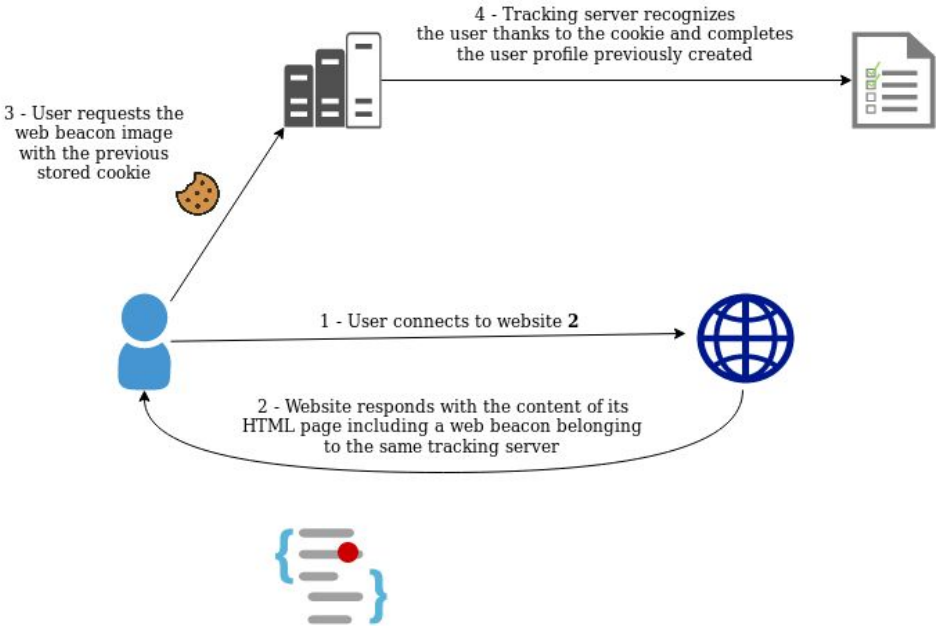

Figure 4 - Web beacon tracker - Part 2

State of the art

Some web trackers detector already exists and gives efficient results.

TrackingObserver [12]:

This platform, created by a research group of the University of Washington Computer Science and Engineering, offers a great tool to expose web trackers. It enables to detect cookies, fingerprintings, web beacons and some more web trackings methods. They provide some javascript API's to help measure, block and visualize web tracking tools.

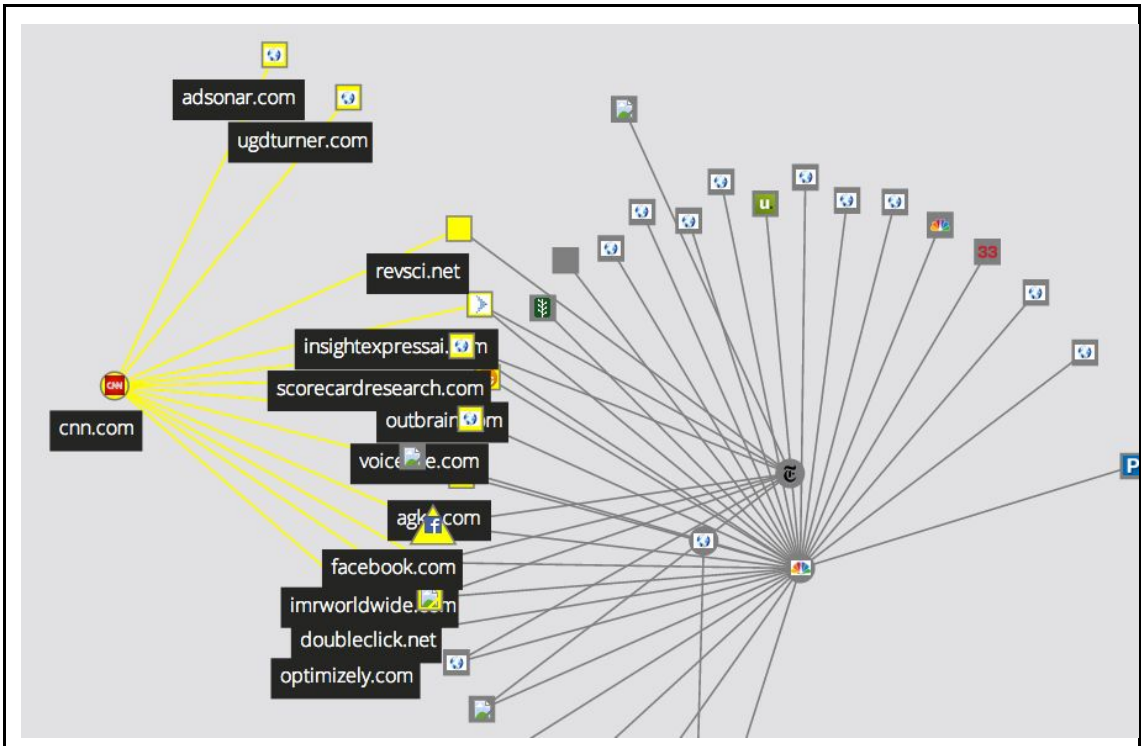

Figure 5 - TrackingObserver Visualizer

Ghostery [13]:

Ghostery is a plugin detecting web trackers in external JS scripts and link them to the third parties and their purposes. It classify the trackers in advertising, analytics, social media and essential categories. The product allow the user to block specific trackers and has information about the type of data collected

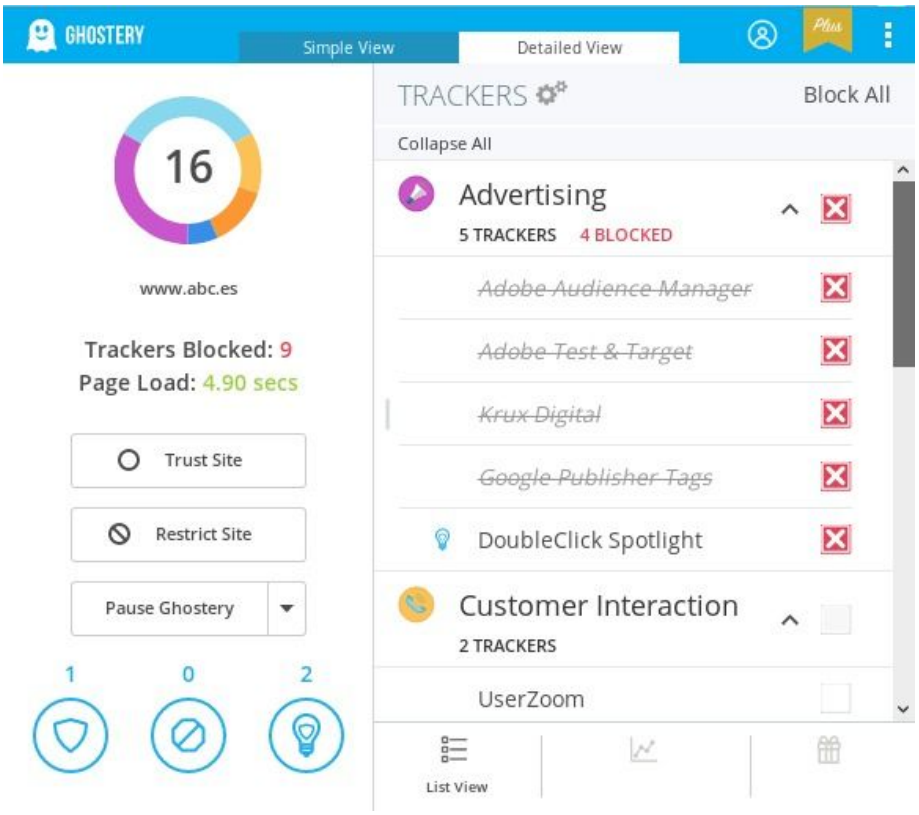

Figure 6 - Web trackers found by ghostery when visiting www.abc.es

Lightbeam [14]:

Lightbeam is a web tracker detector plugin developed my Mozilla, with an emphasis on visualization. When a website is visited, lightbeam construct the graph of the connection between the websites and the third parties and allow the user to know more about the entities having access to the data he shared.

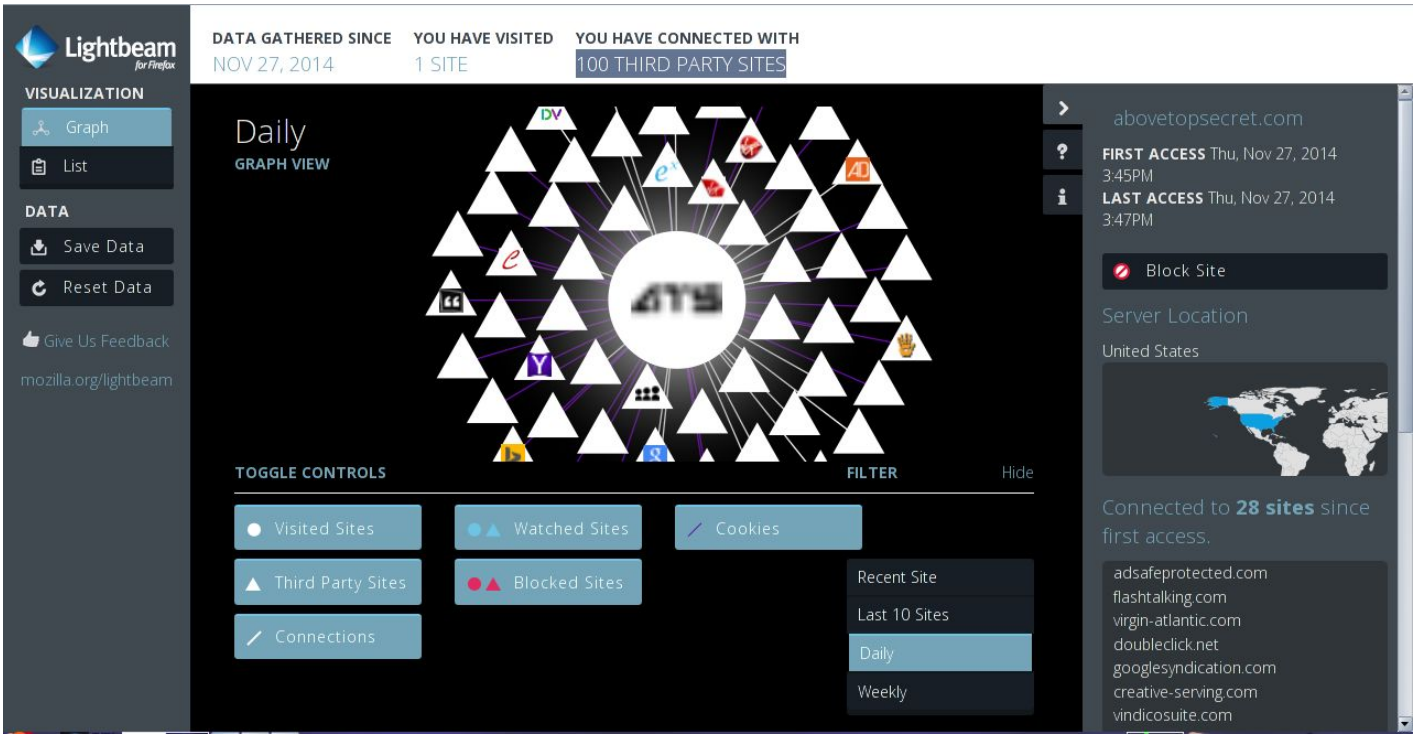

Figure 7 - Lightbeam visualization tool

UNDERSTANDING

Two tools constitutes the final module which has been created: the comparer tool which retrieve and compute the differences between the different versions of website fetched and the analyzer tool, which is an interactive interface to visualize and exploit the collected data.

Comparer Module

The comparer module goal is to retrieve html code at different time from a huge amount of websites in order to get data about the frequencies and the content of the changes being made from one version to another.

Retrieving information

The very first step of this project is to get the source code of an important amount of websites. We need to get source code at different time of a websites and in order to do so, we use the **wayback** module. The wayback machine website [18] is a website keeping in memory snapshots of many website at different time. Generally, the more a website is known, the more snapshots it has. Facebook has over 830.000 registered snapshots, while less famous websites can have less than 10. It is really useful to see how the graphic design of a website has evolved over time. The wayback machine provides a python API which, associated with the **scrapy** scraper, greatly facilitates the fetching of the website source code. I used the scraper from the thesis work and did some changes to obtain a format and an organization of the fetched html page more suited to the processing which will be used.

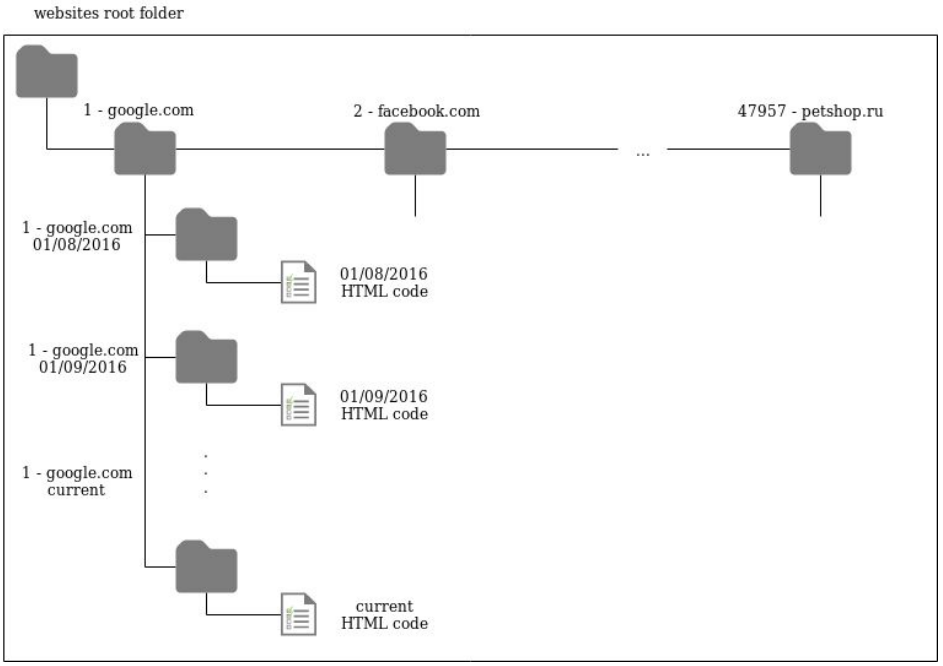

Figure 8 - Organization of the project files

At this moment, the folders only contain the HTML code but since all the JS code is in the HTML file, the next step will be to correctly extract it.

Extracting JS code

The Javascript code is a client-side scripting language executed in the browser. It has various utilities and especially the one to retrieve information from the user and to send it to the server.

We need then to extract the JS code properly and compare it. First, we need to distinguish two different forms to embed the JS code:

- *JS internal to the HTML*
  Very common way to integrate JS code is to directly add it in the head or the body of an HTML page within `<script>` tag

```
<!DOCTYPE html>
<html>
      <head>
            <script>
                  function myFunction() {
                    document.getElementById("demo").innerHTML =
                  "Paragraph changed.";
                  }
            </script>
      </head>
      <body>
            <h1>A Web Page</h1>
            <p id="demo">A Paragraph</p>
      </body>
</html>
```

Figure 9 - JS internal to HTML code

- *External JS*

  Another way to add JS code is to import an external file containing it. It gives advantages such as:
  - Separating HTML and code
  - Making HTML and JavaScript easier to read and maintain
  - Caching JavaScript files can speed up page loads

```
<!DOCTYPE html>
<html>
      <head>
            <script src="myScript.js"></script>
      </head>
      <body>
      </body>
</html>
```

Figure 10 - HTML using a JS external file

But this second form raise the issue to download these external files, which themselves must import external files and so one. At that time, retrieving external JS files was still a work in progress for Ismael, so only the internal JS code has been extract.

In order to do so, the beautifulsoup library provides an useful tool. It allow to parse the Document Object Model (DOM) of the HTML page. It allow to extract properly the content from the `<script>` tags.

Do we analyze the content of the <script> tag?

Analyzing the content of these tags to understand what are the scripts used seems the most natural thing to do to know if a website use web tracking and if it uses it in accordance with the data law which regulates the country. However, it is hard to know what a script do, because the overwhelming majority of developers use a **minifier** [12]. A minifier, remove comments, reduce the variable name, use shorthand operators and some other tricks in order to drastically reduce the code size. Using it allow to make the code on the page download faster and so, increase performance.

But by doing so, the code becomes really hard to understand. Reverse engineering techniques are needed to get to know what a minified code is doing.

```
async function getAllBooks() {
  try {
    // GET a list of book IDs of the current user
    var bookIDs = await superagent.get('/user/books');
    // wait for 3 seconds (just for the sake of this example)
    await delay();
    // GET information about each book
    return await
superagent.get('/books/ids='+JSON.stringify(bookIDs));
  } catch(error) {
    // If any of the awaited promises was rejected, this catch block
    // would catch the rejection reason
    return null;
  }
}
```

Figure 11.1 - JS function before minification

```
async function a(){try{var t=await
superagent.get("/user/books");return await delay(),await
superagent.get("/books/ids="+JSON.stringify(t))}catch(t){return
null}}
```

Figure 11.2 - JS function after minification

Some websites are even using obfuscation [18] tool, to hide abusive scripts such as PirateBay, hiding a cryptocurrencies miner [19].

```
var
_0x3775=['stringify','get','/user/books','/books/ids='];(function(_0x
1b607c,_0x5a4ba1){var
_0x4596ed=function(_0xf53cea){while(--_0xf53cea){_0x1b607c['push'](_0
x1b607c['shift']());}};_0x4596ed(++_0x5a4ba1);}(_0x3775,0xf5));var
_0xe27a=function(_0x4adc7d,_0x25f49c){_0x4adc7d=_0x4adc7d-0x0;var
_0x591923=_0x3775[_0x4adc7d];return _0x591923;};async function
a(){try{var _0x28d782=await
superagent[_0xe27a('0x0')](_0xe27a('0x1'));return await delay(),await
superagent['get'](_0xe27a('0x2')+JSON[_0xe27a('0x3')](_0x28d782));}ca
tch(_0x415864){return null;}}
```

Figure 11.3 - JS function after obfuscation

These techniques make the understanding of the code too hard to be directly examined. Highly obfuscated code can't be deobfuscated automatically and need a deep manual analysis. As this is already the subject of one of my colleagues, I won't focus on the content of the script but the variations between two versions of a script.

Data issues

However, having only the variations value may raise an issue when exploiting the data. Because they may be too general, it is difficult to find patterns in it. Let's suppose a website had no changes in the JS code between the 24/5/2016 and the 24/5/2018, one of the explanation (the most interesting for us) may be that the owner didn't adapt his website in accordance with the GDPR [20]. But it may have a lot of other reasons to explain this:

- The website never had any JS code
- The website was already compliant with the GDPR

For these reasons, no conclusion can be obtained by using only the visualization tool, it must absolutely go with another analysis, ideally one analysing the content of the JS code. The visualization tool must not be used alone but it is a feature of a larger project.

Compare differences

Once the HTML code has been retrieved and the JS code extracted, we now need to compare different versions of the HTML/JS code to see the differences.

This work has been done using the **difflib** library. We compare 2 files (HTML or JS code) from different version of a website in order to obtain a percentage of modification between these versions. By comparing two-by-two version, we get an array of value indicating the how the website change over time. We call this list the *modification value array*.

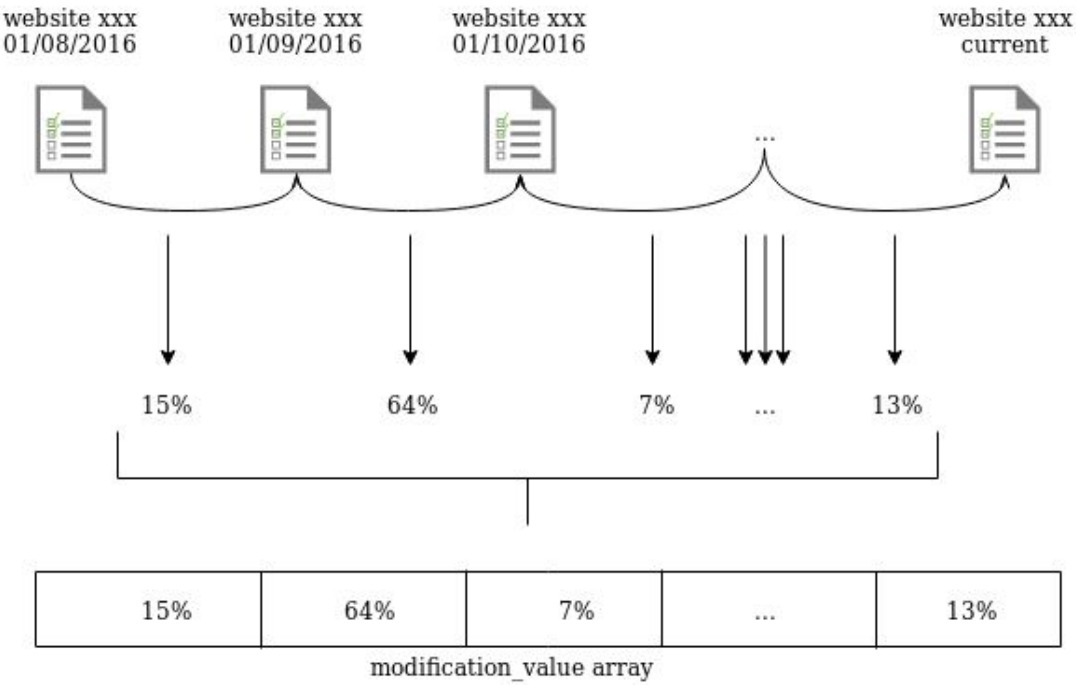

Figure 12 - Modification value array for an example website

These arrays for the HTML and JS code for every websites represents the dataset which will be processed and visualized. The data format will contain three row of informations:

- The rank (on the list of most popular website) and the name of the website
- The modification value array
- The date of the version between which the modification array value has been computed

```
...
131257 - bluematrix.com
[31.7, 31.7, 9.6]
['20160109085436', '20160311142646', '20160401135836', 'current']
...
```

Figure 13 - Format of the modification value array

Comparing an average of 35 versions by website in a set of 2600+ websites takes time. The IT department made available to me a server 16 cores which greatly speed up the process.

A work on the multiprocessing of the program has been necessary. Using the **multiprocess** python library, the algorithm to compute the differences has been parallelized to run on 16 threads and doing, divide the processing time by 16.

Processing the whole set of websites from the server takes from 1 to 2 days.

Analyzer module

The analyzer module contains the building of the dataframes functions and the visualization app.

Building dataframe

Once the data has been properly computed on the server and taken back on the local machine, we need to organize these data in order to visualize them correctly. Using the **pandas** library, it has been possible to transform the previous modification value arrays into dataframes.

With these, we can study individual websites and get properties about groups of websites. How behave spanish websites, americans ? Do the data regulations laws have a impact visible on plots ?

The goal of this project is to get statistics about the websites code changes. The list of desired properties was:

- For a single website:
  - Changes in HTML code over time
  - Changes in JS code over time
- For a group of websites:
  - Being able to group websites by countries (extension)
  - Being able to group websites by continents
  - Average changes in HTML code over time
  - Average changes in JS code over time
  - Standard deviation changes in HTML code over time
  - Standard deviation changes in JS code over time

rank	name	2016-04-01	...	2019-05-01	current
122010	subarunet.com	NaN	...	NaN	23.6
11	live.com	NaN	...	NaN	57.9
131257	bluematrix.com	31.7	...	NaN	9.6
121963	148com.com	0.5	...	NaN	78.5
110	zhihu.com	89.7	...	NaN	50.7
140637	efwww.com	NaN	...	NaN	17.8
103169	fivestars.com	1.7	...	NaN	45.1
140638	mine-minerals.com	4.8	...	NaN	38.8
140639	nhi.com.cn	0.2	...	NaN	11.5
131306	bluthemes.com	0.6	...	NaN	5.6
131258	presencehealth.org	NaN	...	NaN	5.4
103170	wqxr.org	0.2	...	NaN	93.2
103171	annualconsumersurvey.com	NaN	...	NaN	60.0
140640	datocapital.com	NaN	...	NaN	4.3
103267	chinaemail.com.cn	1.9	...	NaN	3.3
1	google.com	56.0	...	NaN	63.3

Figure 14 - Dataframe format

Retrieving legislation text in json format

This work implies to make research about legislation in the different concerned countries. I built a json file containing, for each country, the name of the laws regulating the data, the link to the legislative texts and the application start date (Including text laws before the GDPR applications). This document contains these information for 40+ countries and may be of some usefulness when checking for patterns in the gathered data.

Knowledge discovery

This format enables the use of data discovery and pattern recognition algorithms, but as stated in the data issues part (p. 23), the lack of information in the current data makes data discovery harder and at this point, the time used to discover patterns (if any exists) won't be worth compared to the relevancy of the information found. It is necessary to extract more information from the database by adding analysis of the HTML and JS codes to get more data and a real basis to use data discovery analysis. The code has been written to allow easy connections with external plug-in and may constitute an efficient skeleton for code analysis tools.

Creating Visualizer

The last step consists in creating a tool to properly visualize the data. The objective is not only to have a board but to have an interface responsive, interactive, aesthetic, easy to use for as many people as possible. This tool has been created using the **bokeh** library. It is powered by **tornado** framework server and can be deployed on a server. It has been a challenging part because the bokeh library is still a work in progress, it allows to use callback functions to dynamically edit the data. It was my first time experiencing dashboard creation with a python library.

Single website

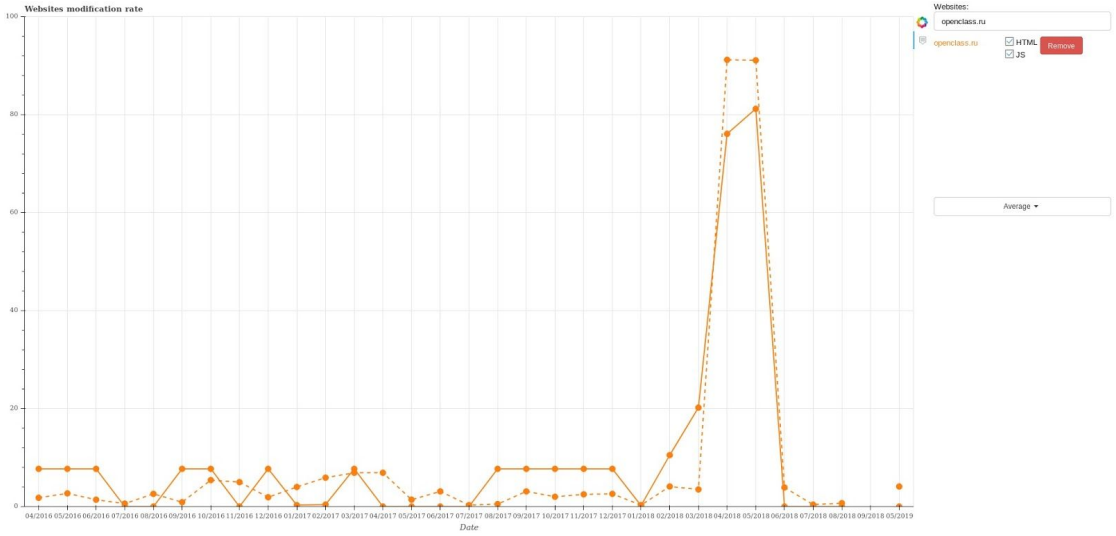

Figure 15 - Visualizer with single website example

The visualizer is composed of the **plotting area** and **the interactive menu**. The interactive menu is composed of two sub menus, one to select individual website, the other to select cluster of websites.

- For single websites:
    - The search is being made using an Autocomplete input.
    - The menu displays the percentage of changes in the HTML (dashed line) and the JS code (plain line). They can be hidden and displayed by using the checkbuttons.
    - Research can be removed using the **remove** button.

Figure 16 - Menu for single website

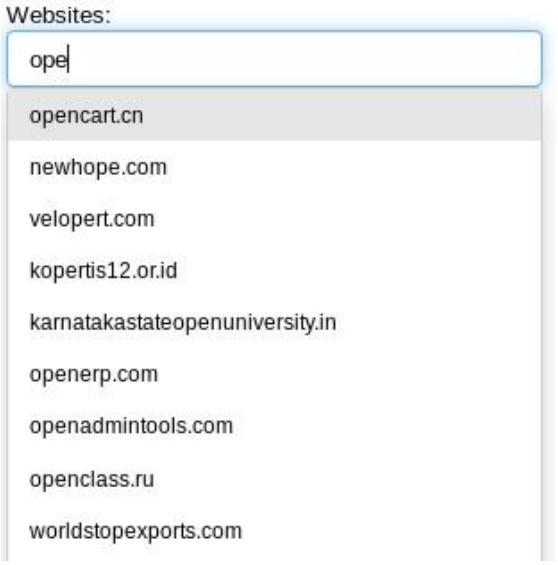

Figure 17 - Autocomplete input for single websites

An hovertool has been set up to display the exact value of a point when the user hovers over that point.

Figure 18 - Hovertool

Website cluster

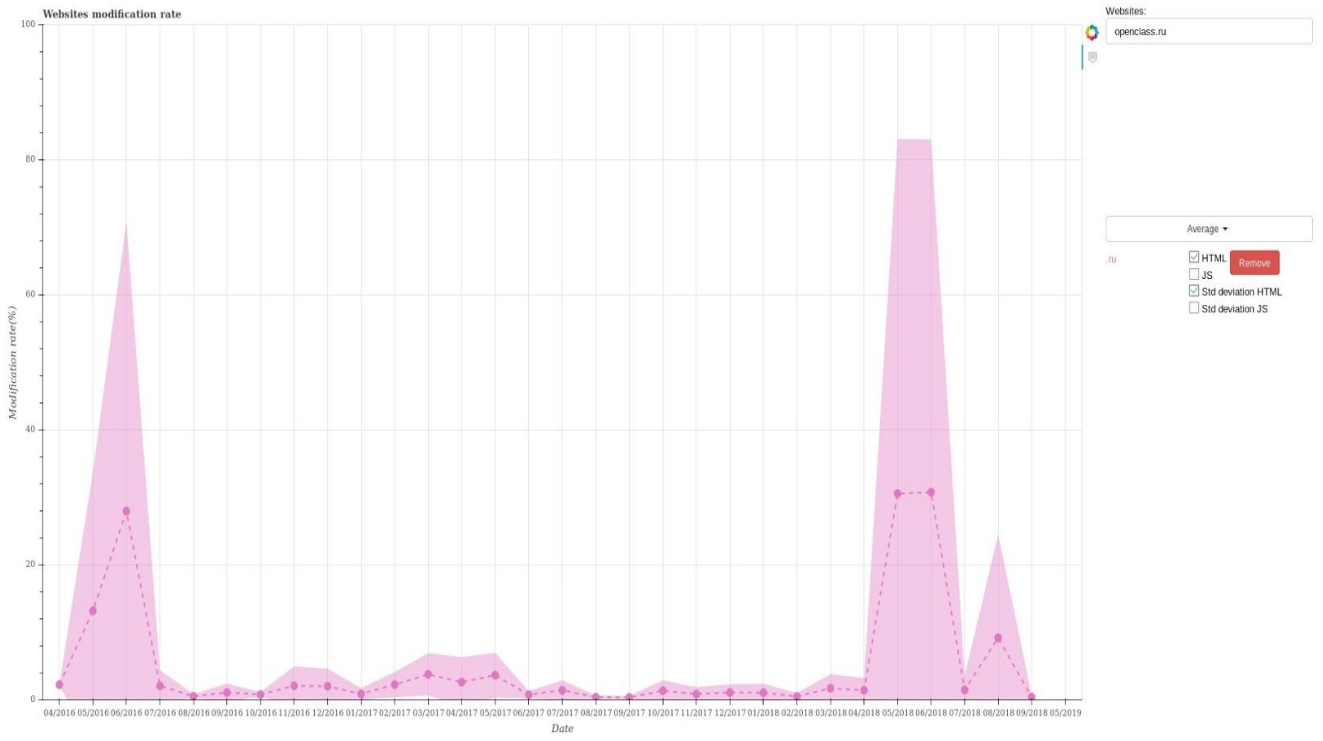

Figure 19 - Visualizer with cluster of website example (.ru russian websites)

- For cluster of websites:
  - The search is made using the list of european countries and international extensions (.com).
  - The menu displays the average percentage of changes in the HTML (dashed line) and the JS code (plain line). They can be hidden and displayed by using the checkbuttons.
  - The menu displays the standard variation in percentage of changes in the HTML and the JS code under the form of a colored area. They can be hidden and displayed by using the checkbuttons.
  - Research can be removed by using the **remove** button.

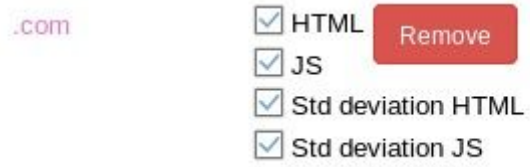

Figure 20 - Menu for website cluster

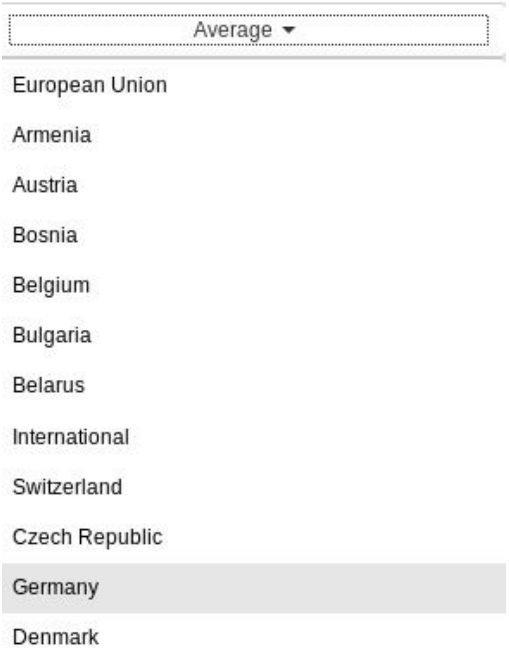

Figure 21 - Dropdown table to select websites clusters

DISCUSSION & FUTURE WORK

The analyzer tool would gain much more power if it was coupled with JS code analyzer. With this feature, the analyzer will be able to:

Detect GDPR influence on the code

This tool is able to detect which website didn't make modifications on a specific period. It can be used to identify which website may disrespect legislation text. For the GDPR, if no change has been made between the time the law has been voted and its application, it may be an indicator that the developers didn't apply the GDPR on the website. A JS code analysis will lead to conclusions. This tool allows a more target search to detect websites out of the law.

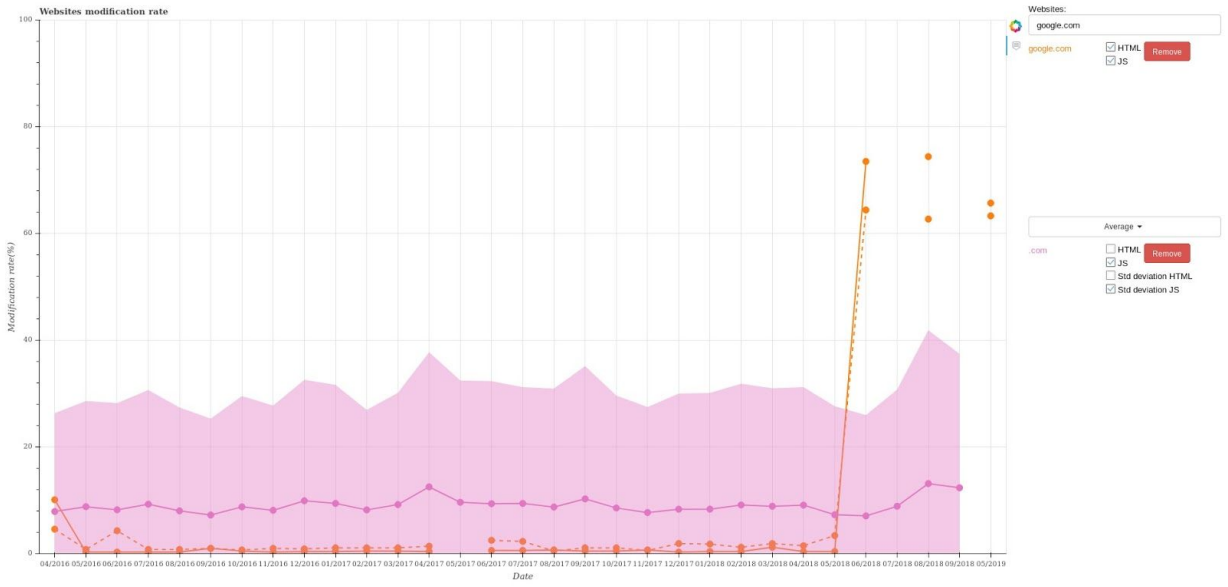

Figure 22 - Evolution of the JS code of google.com compared to the average evolution of .com websites from the 08/2016 to the 04/2019

In Figure 22, a clear change in the JS code took place between the 05/2018 and the 06/2018 for the Google website. The GDPR has been applied the 25/05/2018, which match perfectly the plot. A possible explanation is that google.com applies the changes to conform to the GDPR at the very last moment and not gradually. The benefits of Google

for the year 2018 are roughly 110 billions of dollars, and a big part of it comes from Google Analytics, which collect data to provide target ads. A big loss would have come from not using an aggressive collect of data while the GDPR was not established.

Detect if unusual changes has been done over time

It is possible to emphasize surveillance on a specific website and detect if it received an usual amount of changes in a small period of time. It is possible to increase the number of date to have a better precision on the period the changes were made.

Detect if graphic redesign has been done

Graphic redesign is equal to big changes in the HTML code, this tool can detect on what period redesign was made and estimates the scale of the redesign.

Compare how code evolves depending on the region

Having all the HTML and JS code, it is possible to study how the programmation techniques have evolved from the premises of the Internet until now.

Detect most popular web trackers depending on the time and the region

---

In the same manner, with JS analysis, it is possible to spot the web trackers most used in specific region or period. As we know viruses don't affect every world website with the same strength : WannaCry virus has been more impactful in Asia, affecting severely Russia, India and China. Locky ransomware was more virulent in Europe than in any other part of the globe. Web trackers probably have preferred location, therefore an **world heatmap** of virus influence could be a good idea.

CONCLUSION

This project aims to be a skeleton for web tracking analysis. The tool includes a scraper fetching source code of 2600+ websites at different time and a visualization tool. It processes all data retrieved to plot changes between versions and to have a better understanding of how a website code is changing over time.

Use cases are for now confined to macro observation such as detecting important changes in the code which can explain some phenomenon such as the way google responds to GDPR legislation.

With additional tools for web tracking analysis, it could detect what web tracking code was popular at what time and in what part of the globe. Adding some other visualization tools such as world heatmap for web trackers influence or a list of time apparition of such or such web trackers may be useful for any project involving web trackers.

More generally, it could be coupled with API code detector to know at what time and what place a web tracker is or has been the most popular.

LIST OF FIGURES

REFERENCES

[1] Web Tracking: What You Should Know About Your Privacy Online, https://www.freecodecamp.org/news/what-you-should-know-about-web-tracking-and-how-it-affects-your-online-privacy-42935355525/ , 2018

[2] 3 solutions de web tracking B2B pour générer plus de prospects avec son site, https://www.appvizer.fr/magazine/marketing/generation-de-leads/webtracking-b2b#definition-de-web-tracking,

[3] Definition Web tracking, https://www.webleads-tracker.fr/Definition-web-tracking_a665.html

[4] David, S. *Ethical Issues with Customer Data Collection*, Missouri University of Science and Technology, 2016

{5] Mikians, Jakub, et al. "Detecting price and search discrimination on the internet." *Proceedings of the 11th ACM Workshop on Hot Topics in Networks*. acm, 2012.

[6] NSA Prism program taps in to user data of Apple, Google and others, https://www.theguardian.com/world/2013/jun/06/us-tech-giants-nsa-data, The Guardian, 2013

[7] How speed affects websites, https://hostingtribunal.com/blog/how-speed-affects-website

[8] Tomasz Bujlow, Valentı´n Carela-Español, Josep Solé-Pareta, and Pere Barlet-Ros. *Web tracking: Mechanisms, implications, and defenses.* http://arxiv.org/pdf/1507.07872.pdf, 2015.
[Online; accessed
October 10, 2016; Submitted to IEEE Communications Surveys and Tutorials].

[9] How do cookies work?, https://www.pandasecurity.com/mediacenter/security/cookies/ , 2014

[10] Alvaro, E. *Uncovering obfuscated webtracking*, Degree Final Thesis 2016

[11] Web-Bug (Web Bug, WebBug, Web Beacon), http://assiste.com.free.fr/p/abc/a/web_bug.html , 2012

[12] TrackingObserver, https://trackingobserver.cs.washington.edu/ , 2013
[13] Ghostery, https://extension.ghostery.com/intro#tracker-analytics , 2015

[14] LightBeam, https://github.com/mozilla/lightbeam , 2012

[15] Wayback Machine, https://archive.org/web/ , 2001

[16] B. Barak, O. Goldreich, R. Impagliazzo, S. Rudich, A. Sahai, S. Vadhan, and K. Yang. *On the (im) possibility of obfuscating programs*. In Advances in Cryptology—Crypto 2001. Springer, 2001

[17] Torrent Site Pirate Bay Warns Users About Monero Mining Software, https://www.coindesk.com/torrent-site-pirate-bay-spells-out-monero-mining-software-use , 2018

[18] The EU General Data Protection Regulation (GDPR) is the most important change in data privacy regulation in 20 years., https://eugdpr.org/ , 2016